Le génie pour l'industrie

-

SYS863 - Internet Industriel des Objets

Laboratoire 2

À Montréal, le 08/03/2023

Professeur : Lokman Sboui
Etudiants : Victor Rios, Théodore Ruf

Buts et objectifs du laboratoire :

Dans ce laboratoire nous allons découvrir la couche réseau de l'IIoT en utilisant la plateforme Node-RED. On commencera par explorer les données en local avant d'aborder le protocole MQTT, un protocole de messagerie très populaire pour l'IoT en raison de sa légèreté et de sa faible consommation d'énergie.

Dans un premier temps, nous étudierons l'utilisation des fichiers CSV, précisément en écriture puis leur exploitation sur Excel. Ensuite, nous découvrirons comment afficher les données dans une fenêtre via la bibliothèque GTK+.

De plus, dans ce laboratoire nous nous intéresserons particulièrement à la couche réseau de l'IIoT à travers le protocole MQTT et son implantation avec le broker Mosquitto du Raspberry Pi. On se propose de créer des publieurs/abonnés pour communiquer des informations via le WiFi.

Enfin, on connectera la plateforme de programmation visuelle Node-RED basée sur les flux de données, avec MQTT pour récupérer les informations du Raspberry Pi directement sur la plateforme.

Ce laboratoire nous permettra ainsi de découvrir les fonctionnalités de la couche réseau de l'IIoT et de développer des compétences pratiques en utilisant des outils tels que Node-RED et le protocole MQTT.

2.1 Partie 1 - Sauvegarde locale des données

2.1.1 Les fichiers CSV

Dans cette partie, nous allons sauvegarder les données de la lampe 12V et le thermocouple dans un fichier CSV localement dans la mémoire du Raspberry Pi. Les fichiers CSV (Comma Separated Values) sont un format de fichier de données simple qui utilise des virgules pour séparer les valeurs dans un fichier.

L'utilisation des fichiers CSV permet de stocker les données localement sur le Raspberry Pi, ce qui peut être utile pour des applications IoT qui doivent fonctionner sans accès à Internet ou qui ont besoin d'une sauvegarde locale.

2.1.2 Créer un fichier CSV en C

Nous allons tout d'abord apprendre à créer, écrire et sauvegarder un fichier en C. On utilisera les bibliothèques stdio.h et time.h. Le fichier est créé en utilisant la fonction fopen() et les données sont écrites dans le fichier en utilisant la fonction fprintf(). La fonction strftime() est utilisée pour formater la date et l'heure en un format de chaîne spécifique pour être enregistré dans le fichier CSV. Finalement, le fichier est fermé en utilisant la fonction fclose().

2.1.3 Trouver l'équation de la variation des données en Excel

Dans cette partie, on sauvegarde à intervalle régulier (2 secondes) la température de la lampe de 12 Volts pour une durée de 5 minutes avec une commande PWM constante de 50% dans le but de trouver l'équation de la variation de la température avec le temps de notre système (Lampe+thermocouple).

Pour cela, on peut créer un fichier CSV, récupérer la température grâce à la fonction lecture_SPI(). On règle la commande PWM à 50 %, à l'aide de la fonction intensity().
On peut identifier l'adresse MAC de notre machine en utilisant la commande ifconfig dans le terminal et prendre celle du eth0. Enfin, on peut écrire l'ensemble de ces données dans le fichier CSV avec la fonction fprintf().

Les données sont enregistrées dans un excel dans l'ordre suivant : date et heure, adresse MAC de la raspberry, numéro du poste sur lequel on travaille et la température toutes les deux secondes. Dans notre programme nous récupérons 150 valeurs de la température au total, soit une toutes les deux secondes pendant 5 minutes à l'aide d'une boucle for :

```
for (int cpt = 0; cpt < 150; cpt ++)
```

3	2020-01-09 21:56:16	b8:27:eb:ec:58:7b	Poste 11	55,5
4	2020-01-09 21:56:18	b8:27:eb:ec:58:7b	Poste 11	55,8
5	2020-01-09 21:56:20	b8:27:eb:ec:58:7b	Poste 11	55,8
6	2020-01-09 21:56:22	b8:27:eb:ec:58:7b	Poste 11	55,8
7	2020-01-09 21:56:24	b8:27:eb:ec:58:7b	Poste 11	56
8	2020-01-09 21:56:26	b8:27:eb:ec:58:7b	Poste 11	56
9	2020-01-09 21:56:28	b8:27:eb:ec:58:7b	Poste 11	56,2
10	2020-01-09 21:56:30	b8:27:eb:ec:58:7b	Poste 11	56,2
11	2020-01-09 21:56:32	b8:27:eb:ec:58:7b	Poste 11	56,2
12	2020-01-09 21:56:34	b8:27:eb:ec:58:7b	Poste 11	56,2
13	2020-01-09 21:56:36	b8:27:eb:ec:58:7b	Poste 11	56,2
14	2020-01-09 21:56:38	b8:27:eb:ec:58:7b	Poste 11	56,5
15	2020-01-09 21:56:40	b8:27:eb:ec:58:7b	Poste 11	56,5
16	2020-01-09 21:56:42	b8:27:eb:ec:58:7b	Poste 11	56,5
17	2020-01-09 21:56:44	b8:27:eb:ec:58:7b	Poste 11	56,5
18	2020-01-09 21:56:46	b8:27:eb:ec:58:7b	Poste 11	56,5
19	2020-01-09 21:56:48	b8:27:eb:ec:58:7b	Poste 11	56,8
20	2020-01-09 21:56:50	b8:27:eb:ec:58:7b	Poste 11	56,8
21	2020-01-09 21:56:52	b8:27:eb:ec:58:7b	Poste 11	56,8

Table 1. Tableau excel contenant les données enregistrées en fichier CSV

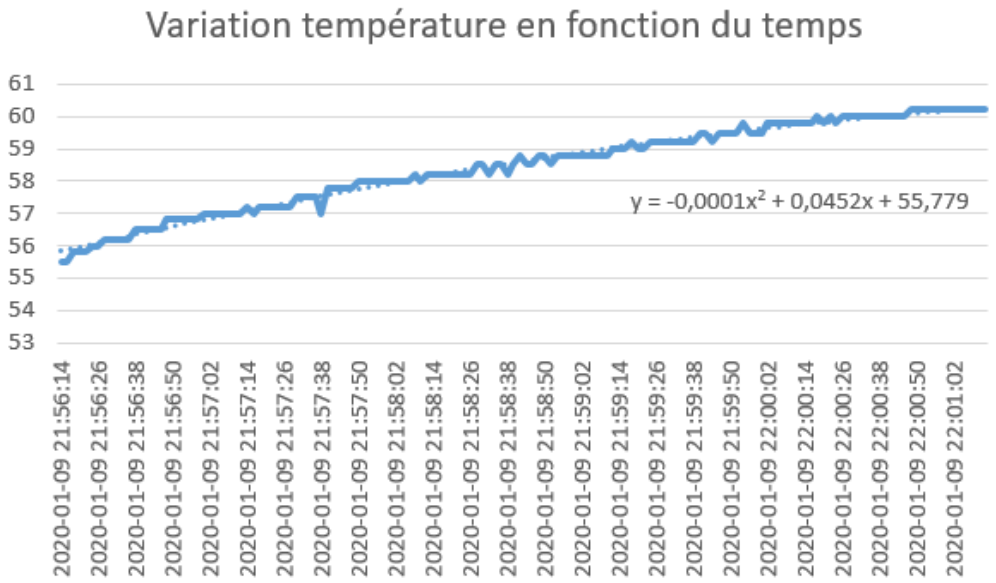


Figure 1. Courbe de l'évolution et de tendance de la température par rapport au temps sur les 5 minutes.

Après avoir tracé la courbe de la température en fonction de la date, nous avons tracé une courbe de tendance. Cette dernière est un polynôme du second degré car il correspond parfaitement à l'allure de la variation de la température par rapport au temps.

On peut ainsi extraire l'équation de la température en fonction du temps : $y = -0.0001x^2+0.0452x + 55.79$

On constate que la température augmente dans le temps car la consigne est constante.

2.1.4 Variation de PID en fichier CSV

Dans cette partie, nous sauvegardons toujours à intervalle de deux secondes la température de la lampe mais cette fois pour une durée de 15 minutes avec une commande PID. Après avoir fixé les valeurs des gains en Kp = 10, Ki = 10, Kd = 0.2, la consigne à la température initiale + 3 degrés. Deux nouvelles données sont enregistrées dans le fichier csv : Consigne et Commande en %.

De la même manière que précédemment, nous lisons la température, mais cette fois-ci nous utilisons une commande PID pour contrôler la température à l'aide de la fonction commande_PID().

Les données sont stockées de la même manière que dans la partie 2.1.3 dans l'ordre suivant : date et heure, adresse MAC de la raspberry, numéro du poste sur lequel on travaille et la température toutes les deux secondes, la consigne et la commande.

	Date et heure	Adresse MAC	Emplacement	Temperature	Consigne	Commande en pourcent
2	2020-01-09 23:36:2	b8:27:eb:ec:58:7t	Poste 11	36,2	39	27
3	2020-01-09 23:36:2	b8:27:eb:ec:58:7t	Poste 11	36,2	39	45
4	2020-01-09 23:36:3	b8:27:eb:ec:58:7t	Poste 11	36,2	39	63
5	2020-01-09 23:36:3	b8:27:eb:ec:58:7t	Poste 11	36,5	39	78
6	2020-01-09 23:36:3	b8:27:eb:ec:58:7t	Poste 11	36,8	39	92
7	2020-01-09 23:36:3	b8:27:eb:ec:58:7t	Poste 11	37	39	100
8	2020-01-09 23:36:3	b8:27:eb:ec:58:7t	Poste 11	37,2	39	100
9	2020-01-09 23:36:4	b8:27:eb:ec:58:7t	Poste 11	37,8	39	100
10	2020-01-09 23:36:4	b8:27:eb:ec:58:7t	Poste 11	38,2	39	100
11	2020-01-09 23:36:4	b8:27:eb:ec:58:7t	Poste 11	38,5	39	100
12	2020-01-09 23:36:4	b8:27:eb:ec:58:7t	Poste 11	38,8	39	100
13	2020-01-09 23:36:4	b8:27:eb:ec:58:7t	Poste 11	39	39	98
14	2020-01-09 23:36:5	b8:27:eb:ec:58:7t	Poste 11	39,2	39	96
15	2020-01-09 23:36:5	b8:27:eb:ec:58:7t	Poste 11	39,8	39	88
16	2020-01-09 23:36:5	b8:27:eb:ec:58:7t	Poste 11	40	39	80
17	2020-01-09 23:36:5	b8:27:eb:ec:58:7t	Poste 11	40,2	39	71
18	2020-01-09 23:36:5	b8:27:eb:ec:58:7t	Poste 11	40,2	39	62
19	2020-01-09 23:37:0	b8:27:eb:ec:58:7t	Poste 11	40,5	39	51
20	2020-01-09 23:37:0	b8:27:eb:ec:58:7t	Poste 11	40,5	39	41
21	2020-01-09 23:37:0	b8:27:eb:ec:58:7t	Poste 11	40,5	39	31

Table 2. Tableau excel contenant les données enregistrées en fichier CSV

Cette fois, nous récupérons 450 valeurs de la température au total, soit une toutes les deux secondes pendant 15 minutes à l'aide de la boucle for.

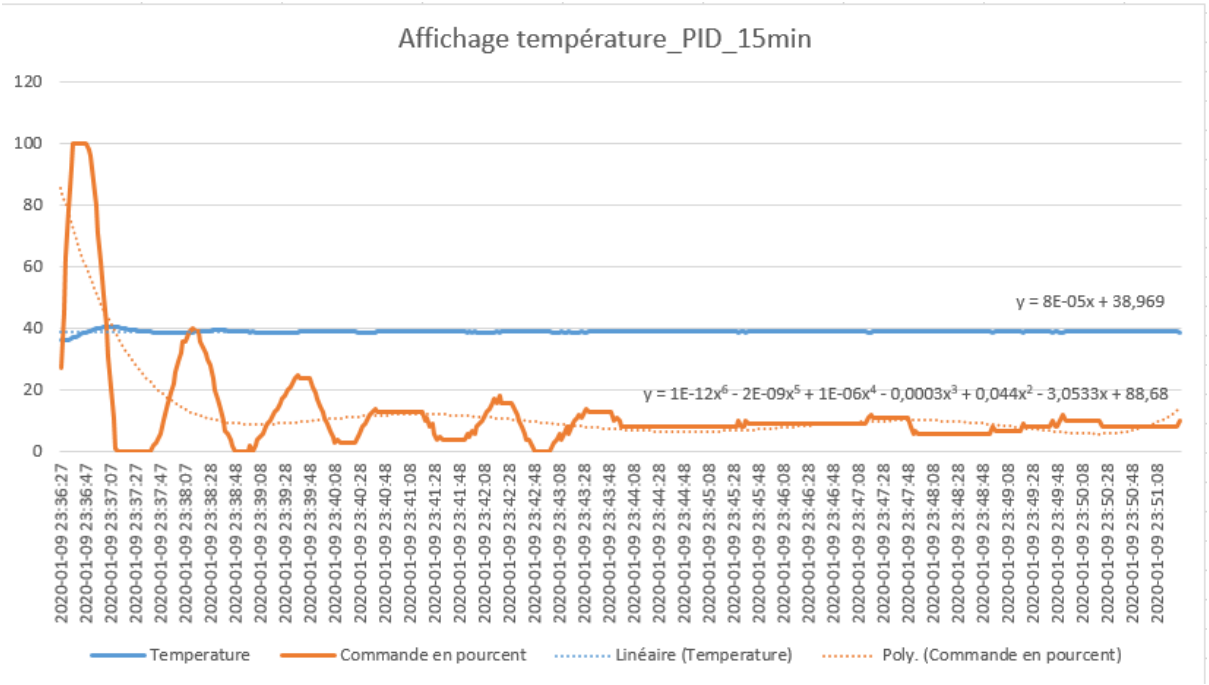


Figure 2. Courbes de l'évolution et de tendance de la température et de la commande par rapport au temps

Après avoir tracé les courbes de la température en fonction de la date, nous avons tracé deux courbes de tendance :

- La première qui suit la courbe de la température est un polynôme est une fonction affine y = 8e-5x + 38,969.,
- La deuxième qui suit la courbe de commande (%) est un polynôme de degré 6 d'équation y = 1e-12x^6 - 1e-9x^5 - 1e-6x^4 - 0,0003x-^3 - 3,0533x + 88,68.

On remarque que la commande s'adapte pour conserver la température consignée et oscille avec un rebond de plus en plus faible.

2.2 Partie 2 - Affichage dans une fenêtre

2.2.1 La bibliothèque GTK+

La bibliothèque GTK+ (GIMP Toolkit) est une bibliothèque logicielle open source qui permet de créer des interfaces graphiques utilisateur (GUI) pour les applications. GTK+ est conçu pour être flexible, rapide et facile à utiliser. Il fournit une variété de widgets pour la construction d'interfaces graphiques, tels que des boîtes, des boutons, des entrées de texte et des étiquettes. GTK+ est développée en langage de programmation C et utilise des widgets (éléments d'interface graphique) pour créer des interfaces utilisateur.

2.2.2 Exemple de fenêtre en C

L'intérêt de cette partie est de se familiariser avec la bibliothèque GTK+. Pour ce faire, on affiche la température de la lampe dans une fenêtre toutes les secondes.

On appelle la fonction g_timeout_add() qui appelle toutes les secondes la fonction *update_temp* qui récupère la nouvelle température de la lampe. Ce dernier actualise à son tour la nouvelle valeur de la température dans la fenêtre GTK à l'aide de la fonction lecture_SPI(). En premier paramètre de g_timeout_add() est affectée la valeur "500" qui a pour effet d'actualiser toutes les secondes la température.

2.3 Partie 3 - Mosquitto et le MQTT

2.3.1 WiFi et Raspberry Pi

Nous travaillerons dans cette partie dans la couche réseau, au niveau du WiFi pour la communication avec l'extérieur. Comme tout nœud sur le WiFi, le Raspberry Pi possède un identifiant qu'il acquiert lorsqu'il se connecte, l'adresse IP. Comme on travaille dans le standard IPv4, l'adresse IP est constituée d'une suite de 4 nombres (de type char 8 bits) séparés par des points. Cette adresse IP va nous servir dès que nous utiliserons un équipement qui devra accéder à un serveur ou au broker MQTT du Raspberry Pi.

2.3.2 Mosquitto et le MQTT

Dans cette première partie, nous expérimentons le protocole MQTT avec le broker Mosquitto installé dans le Raspberry Pi. Mosquitto est un broker MQTT open source qui permet la communication entre différents périphériques IoT en utilisant le protocole MQTT. MQTT (Message Queuing Telemetry Transport) est un protocole de messagerie léger qui permet de transmettre des données entre différents périphériques connectés à un réseau, même dans des environnements à bande passante limitée.

2.3.4 Publieurs et abonnés

MQTT suit un modèle de publication/abonnement (pub/sub) où les dispositifs connectés sont soit des publieurs, soit des abonnés. Les publieurs publient des messages sur des sujets spécifiques, et les abonnés reçoivent les messages publiés sur les sujets auxquels ils sont abonnés. Ce modèle permet une communication efficace entre les dispositifs connectés en évitant les problèmes de connectivité persistante.

Les clients Mosquitto peuvent être soit des publieurs qui envoient des messages à des sujets spécifiques, soit des abonnés qui s'abonnent à des sujets pour recevoir les messages publiés sur ces sujets. Les sujets dans Mosquitto peuvent être organisés hiérarchiquement pour permettre une gestion plus efficace et flexible des messages publiés et reçus.

2.3.5 MQTT dans un programme en C

a) Lecture de la température

Dans cette partie, nous utiliserons un publieur MQTT pour publier chaque 2 secondes la température de la lampe de 12 Volts. De plus, on implante également un abonné MQTT pour qu'il affiche à intervalle régulier la température de la lampe de 12 Volts mesurée envoyée par le publieur. Les deux programmes devront donc fonctionner

simultanément et s'arrêter à l'appui sur le bouton rouge. Enfin, la valeur reçue par l'abonné sera affichée dans une fenêtre GTK.

Pour l'ensemble du système :

- MQTT_HOSTNAME "localhost" // Adresse IP du broker
- MQTT_PORT 1883 // Port applicatif pour MQTT

Publieur : Pour ce faire, on récupère la température à l'aide de la fonction lecture_SPI(). On crée un thread publish dans lequel on se connecte au serveur Mosquitto (Broker) qui va faire le lien entre l'abonné et le publieur avec mosquitto_connect() . Ensuite, on publie la température mise à jour sur le topic "temp".

Abonné : Dans ce programme, on utilise directement le thread de la fenêtre GTK que l'on vient de créer pour écouter le broker sur le topic choisi de façon périodique. Ainsi, après avoir configuré l'abonné et l'avoir connecté au broker, on le subscribe au topic "temp". On utilise alors la fonction mosquitto_message_callback_set() pour appeler périodiquement la fonction my_message_callback() qui s'occupera elle de vérifier si un nouveau message est parvenu et d'afficher sa valeur si il correspond bien entendu au topic "temp". Enfin, on updatera la valeur sur la fenetre GTK chaque seconde avec la fonction update_temp() et gtk_label_set_text().

Le programme s'arrête lors de l'appui du bouton rouge (BTN2) grâce à une configuration dans l'abonné et du pooling dans le publieur. En effet, à chaque fois que la température est actualisée sur la fenêtre GTK par update_temp à toutes les secondes lorsque que g_timeout_add est appelé, on vérifie une condition de pooling sur le bouton. Cette dernière permet de vérifier si il y a appui sur le bouton au quel cas le programme s'arrête, sinon il continue de tourner normalement.

b) Ajout de la commande PID

On crée un nouveau programme, qui cette fois, met en relation :

- la commande PID faite au laboratoire 1 pour en faire un publieur de la température du thermocouple (QoS de 0).
- La consigne de température (QoS de 0).
- Le pourcentage PWM calculé (QoS de 0).
- Et l'état désiré des deux DEL (QoS de 2).

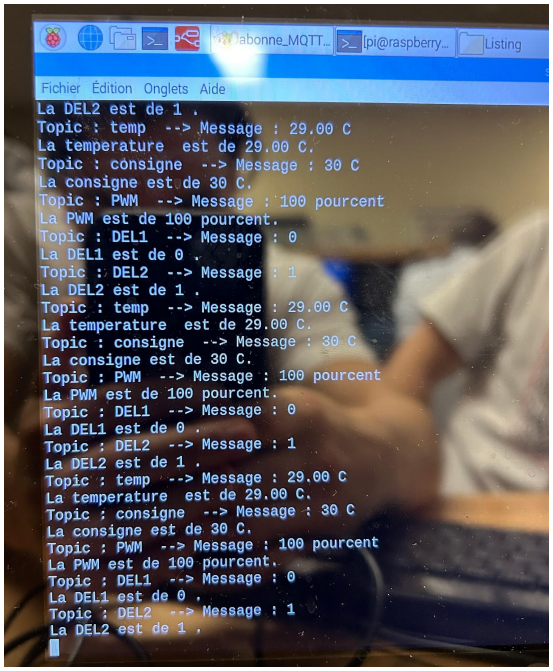
Chacun de ces paramètres seront publiés sur des topics propre à chacun, et réceptionnés par l'abonné sur ces mêmes topics.
On fixe les valeurs des gains du PID qui resteront non modifiés, tels que les thermostats électroniques disponibles sur le marché. Ces valeurs sont Kp = 10, Ki = 10, Kd = 0.2.

Publieur : Le fonctionnement de ce publieur est similaire au précédent, à la différence que l'on va récupérer l'ensemble des informations des capteurs et publier sur les différents topics les informations correspondantes.

- On récupère la température à l'aide de la fonction lecture_SPI(). On crée un thread publish dans lequel on se connecte au serveur Mosquitto (Broker) qui va faire le lien entre l'abonné et le publieur avec mosquitto_connect() . Ensuite, on publie la température mise à jour sur le topic “temp”.
- On récupère la consigne qui est seulement une définition. Ainsi la consigne est publiée et accessible par les abonnés sur le topic “MQTT_TOPIC2, soit, “consigne”.
- On récupère le pourcentage PWM calculé par la fonction commande_PID() par rapport aux gains fixés, la consigne et la température actuelle. Cette valeur est ensuite transformée en pourcentage qui sera celle publiée et mise à jour sur le topic “MQTT_TOPIC3”, soit, “PWM”.
- On récupère l'état des deux DEL avec la fonction bcm2835_gpio_lev() de la librairie bcm2835. Leurs états sont ensuite publiés et mis à jour sur les topics “MQTT_TOPIC4” et “MQTT_TOPIC5” correspondant respectivement au DEL1 et DEL2.

Abonné : L'abonné est un peu plus compliqué que le précédent. En effet, on va également utiliser directement le thread de la fenêtre GTK que l'on vient de créer pour écouter le broker sur tous les topics subscribed de façon périodique. Ainsi, après avoir configuré l'abonné et l'avoir connecté au broker, on le subscribe aux topics voulues. On appelle périodiquement my_message_callback(), qui lit le message reçu par le broker, on vérifie si le message contient le topic subsribed, dans ce cas, on affiche la valeur reçue dans le terminal. Enfin, on updatera la valeur sur la fenetre GTK chaque seconde avec la fonction update_temp() et gtk_label_set_text().

Comme précédemment, le programme s'arrête lors de l'appui du bouton rouge (BTN2) grâce à une configuration dans l'abonné et du pooling dans le publieur. En effet, à chaque fois que la température est actualisée sur la fenêtre GTK par update_temp à toutes les secondes lorsque que g_timeout_add est appelé, on vérifie une condition de pooling sur le bouton. Cette dernière permet de vérifier si il y a appui sur le bouton au quel cas le programme s'arrête, sinon il continue de tourner normalement.



```
La DEL2 est de 1 .
Topic : temp  --> Message : 29.00 C
La temperature  est de 29.00 C.
Topic : consigne  --> Message : 30 C
La consigne est de 30 C.
Topic : PWM  --> Message : 100 pourcent
La PWM est de 100 pourcent.
Topic : DEL1  --> Message : 0
La DEL1 est de 0 .
Topic : DEL2  --> Message : 1
La DEL2 est de 1 .
Topic : temp  --> Message : 29.00 C
La temperature  est de 29.00 C.
Topic : consigne  --> Message : 30 C
La consigne est de 30 C.
Topic : PWM  --> Message : 100 pourcent
La PWM est de 100 pourcent.
Topic : DEL1  --> Message : 0
La DEL1 est de 0 .
Topic : DEL2  --> Message : 1
La DEL2 est de 1 .
Topic : temp  --> Message : 29.00 C
La temperature  est de 29.00 C.
Topic : consigne  --> Message : 30 C
La consigne est de 30 C.
Topic : PWM  --> Message : 100 pourcent
La PWM est de 100 pourcent.
Topic : DEL1  --> Message : 0
La DEL1 est de 0 .
Topic : DEL2  --> Message : 1
La DEL2 est de 1 .
```

Figure 3. Capture d'écran de la réception des topics par l'abonné

Utilisation d'une application MQTT mobile : MQTTool

On établit la connexion avec le broker MQTT et on s'abonne à la température sur le topic "temp", tout en lançant le publieur en parallèle.

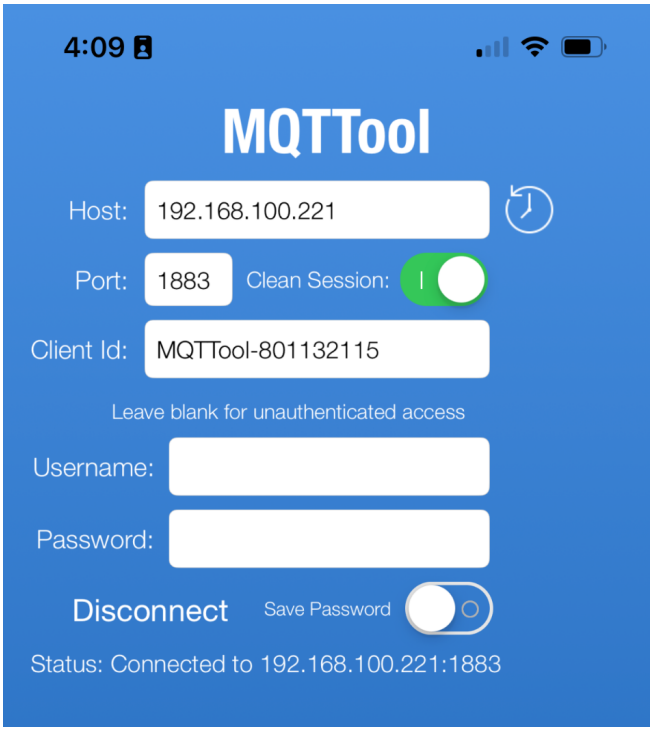


Figure 4. Configuration de la connexion MQTT

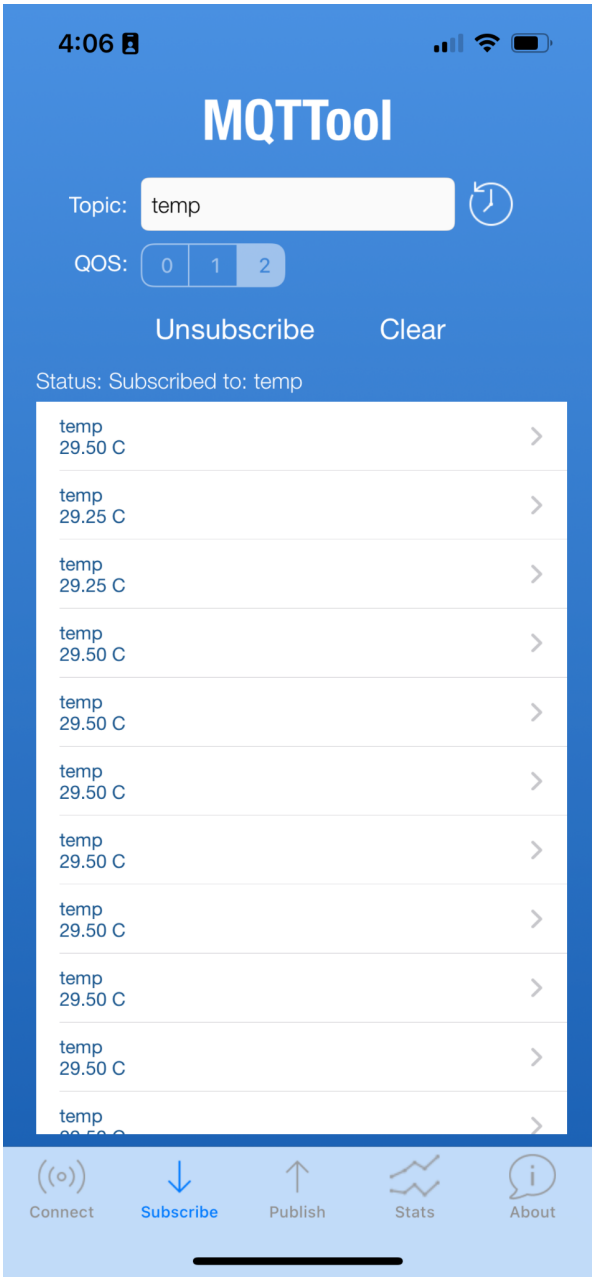


Figure 5. Réception de la température par l'abonné sur le topic "temp"

On configure l'outil de la même manière que dans le code en utilisant ici l'adresse IP du Raspberry Pi (pour accéder à un serveur et/ou broker du la Raspberry Pi) et on observe à la manière de la réception de tous les topics dans le terminal, la réception du topic "temp" auquel on s'est abonné.

2.4 Partie 4 - Node-Red

Dans cette partie, on combine l'environnement HMI Node-Red avec le MQTT. Le programme de commande PID avec MQTT ayant été programmé dans la section précédente, il nous restait simplement à introduire l'utilisation de l'interface Node-Red. Node-Red permet de passer outre les problèmes des applications MQTT (en particulier dans l'environnement iOS). Il est connu pour son interface et son accessibilité via un navigateur WEB.

2.4.1 Lancement de Node-Red

Node-Red est un environnement de programmation visuelle basé sur Node.js qui permet de connecter des appareils et des services IoT. Node-Red utilise une interface graphique pour représenter le flux de données entre les différents appareils, services et applications. Pour combiner Node-Red avec MQTT, nous allons utiliser le nœud MQTT de Node-Red, qui permet de connecter Node-Red à un broker MQTT.

On peut le lancer directement sur le Raspberry Pi sur l'IP : 127.0.0.1 et sur le port 1880 pour l'application Node-Red. On ouvre alors un flow et on est prêt à introduire nos nouveaux nœuds.

2.4.2 Premier programme Node-RED

Dans cette partie, on découvre comment utiliser Node-RED, les noeuds inject, debug et la fonctionnalité Deploy permettant de lancer l'exécution et le débogage du workflow.

De plus, le nœud mqtt in sera utile dans ce projet pour s'abonner à des topics MQTT. On doit sélectionner le serveur MQTT, définir le topic auquel on s'abonne, la qualité de service (0, 1 ou 2) et le type de sortie. Le nœud mqtt out permet de publier des messages vers le broker MQTT. Dans ce projet, il servira à fournir les consignes de température et de niveau à notre réservoir virtuel. On doit sélectionner le serveur MQTT, définir le topic sur lequel on publie, la qualité de service (0, 1 ou 2) et indiquer si la dernière publication doit être conservée par le broker (retain).

2.4.3 Les Noeuds de Node-RED

Dans cette partie, on s'attache à afficher la température mesurée du thermocouple, la consigne et la commande sur l'interface Node-Red. La structure est la suivante :

- Configuration d'un nœud mqtt out pour permettre de publier des messages vers le broker MQTT. On définit le serveur localhost et le port 1883. Celui-ci gèrera les messages envoyés par le publieur précédent.
- Configuration de 3 nœuds mqtt in pour s'abonner aux topics MQTT "temp", "consigne" et "PWM" et récupérer leur valeur à chaque envoi par le publieur et réception par le broker.
- Enfin, la configuration de 3 nœuds Debug pour afficher la charge utile (payload) du signal mqtt envoyé par les liens tissés entre les nœuds.

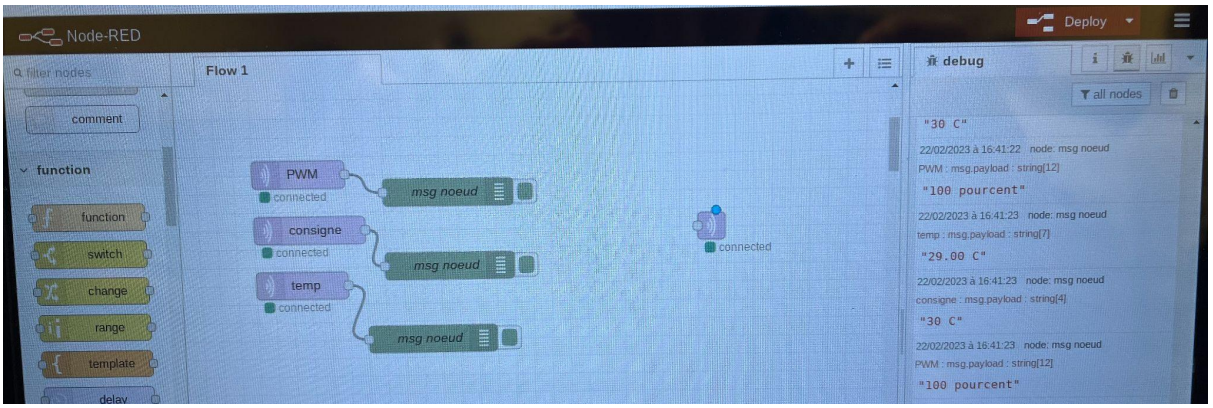


Figure 6. Flow montrant la réception MQTT des topics et leur affichage dans le cadre debug

<u>Bonus</u> :

Nous avons tous les deux apprécié la richesse de ce laboratoire, que ce soit la découverte des sauvegardes locales des données dans les fichiers csv, des bibliothèques GTK+ offrant la possibilité d'afficher des fenêtres graphiques, le broker Mosquitto et le protocole de communication MQTT largement utilisé dans l'industrie ou encore Node-Red connu pour son interface intuitive et ses performances. On a pu élargir nos connaissances en matière de possibilités de la Raspberry.

Cependant, on a eu un peu de mal à comprendre le fonctionnement du thread gtk_main() et la façon dont sortir de cette boucle. Le sujet contient beaucoup de texte notamment sur la partie publieurs/abonnés, ce qui a pour effet de nous perdre un peu. De plus, les listings qui sont présentés ne sont pas utilisés, c'est intéressant mais quand même un peu long à traiter en détail alors qu'il faut aussi vite avancer dans le laboratoire.